[illegible] [illegible]

LEVEL II (12)

Effects of Special Purpose Hardware in Scientific Computation With Emphasis on Power System Applications

Diane M. Detig

November 1980

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

14 CMU-CS-81-108

Effects of Special Purpose Hardware in Scientific Computation With Emphasis on Power System Applications

10 Diane M. Detig**

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

11 Nov[redacted] 12 19

* This research was supported in part by the Office of Naval Research under Contract N00014-80-C-0236. Part of this research was performed during the summer of 1980, while the author was at TRW-Array Processors, San Jose, CA.

** Author's present address: Cray Research, Inc., 1440 Northland Drive, Mendota Heights, MN 55120.

Abstract

Several uses for special purpose numerical hardware for scientific applications have recently been proposed [1, 3, 6]. However, few of those innovations have yet been implemented. We discuss some of the characteristics essential for an application to effectively use a special purpose processor. We examine some power system applications and argue that their present form is not suitable for systems that include special purpose processors.

Table of Contents

1. Introduction

Organizations that deal with real-time compute-bound problems or data compute-bound problems, or suffer from not enough "cycles for the dollar" are looking towards new architectures to satisfy their needs. A thorough study of these architectures would examine all feasible computer configurations, selecting the most efficient combination of architecture and algorithm. Since many features of a system are difficult or expensive to simulate, building a prototype system may be faster and cheaper than any accurate simulation. Often the only true benchmarks are measurements of the complete, working system. In order to avoid the huge expense incurred when examining alternative architectures, we introduce requirements that facilitate determining if an application can effectively use an architecture involving a special purpose device.

Special purpose hardware devices are designed to perform a small number of specific tasks. Usually they are hardwired, fast, inexpensive, and commercially available. Examples of such devices are memories and floating point accelerators. Most of the observations made in this paper will be made with respect to array processors but will be equally applicable to other special purpose hardware devices. An array processor is a form of special purpose processor. It is a high speed, programmable, peripheral device designed for performing numerical calculations on arrays of data. Often the ALU of an array processor is made up of many functional units, so that operations can be performed in parallel. The functional units are sometimes pipelined. Data and program stores are usually small and separate from each other.

The code used by most power system simulations was written for use on a standard von-Neumann machine. The applications tend to be very large: as many as a quarter of a million lines of Fortran code using one to ten megabytes of data memory. Human interface is used extensively in many applications: as much as 15% of the total compute time may be spent interfacing with the user [5]. This code has been evolving over the last twenty years and has not been optimized for any particular machine. The systems are written in a high level language, allowing low development time, low maintainence costs, and all the frills of management.

The observations made in this paper will be with respect to the power system applications, but will apply to many other scientific applications as well. Like most scientific computations, power system applications require high throughput of diverse arithmetic operations [5], high precision, and a wide range of values. Not all computations performed are numerical; many logical operations are required. Pointer manipulations are heavily employed in sparse matrix routines. There are very few compute-intensive routines. The algorithms are still developing and the computations required are not fixed.

Much of the information used in this paper was obtained in conversations with people in the power system industry. The general concensus is that there are no absolute percentages of computation time that can be associated with each type of operation. Even though there is no single description of their computational requirements, the discussion in this paper applies to most of the industry.

We introduce requirements of an application necessary for efficient use of special purpose processors. The requirements include: small compute-intensive routines, small communication, parallel functions, regular memory access, and small data memory. We examine the power system computations with respect to these features with mention of the algorithms which may develop and the charcteristics of special purpose processors which are most desirable for scientific computation. We conclude with a summary of the usefulness of special purpose processors for power system applications.

2. Requirements of an Application for Effective Use of Special Purpose Processors

Any computer application includes algorithms and tasks which we will call the environment of operation. These include maintaining system data, displaying results, maintaining and developing code, and operating machines. Certain characteristics of the environment may limit the usefulness of special purpose processing. This section will establish certain requirements of the algorithms and the environment that are necessary in order to effectively use special purpose processors.

2.1 Requirements Arising from Multiprocessor Issues

When there is more than one machine employed, as in the case of the use of special purpose processors together with a host system, the usefulness of all processors in the solution, the amount of time spent communicating between machines, and the configuation of the system are all issues of concern. In particular:

- The algorithm must fit well into the system architecture, i.e. must have useful work for both the special purpose processor and the host to perform in the correct proportion of time.
- The algorithms must be structured so that there is very little data to be communicated and a small interface time is feasible.
- Unless there is little manipulation of large databases, very slow disk input and output may be a bottleneck of an algorithm.
- The algorithm must use special purpose processors to execute most of the computation.

2.2 Requirements Arising from Hardware Issues

Information about hardware features (such as timing diagrams) might be the key to an efficient implementation. Since it is better to exploit the architecture than to fight it, we must be sure that it will nicely fit the algorithm. In particular:

- Storage requirements must be very small. There is little room in the special purpose processor for code or data. The intent of the special purpose processor is not mass storage but mass execution of arithmetic operations.

- The algorithm must account for the limited precision of the machine.

- We must express our computation as many operations with only a few memory accesses and we must keep our memory access mechanisms simple. Memory devices are becoming cheaper and faster but those with cycle times comparable to the state-of-the-art ALU are either relatively expensive or not yet commercially available.

- The algorithm should employ only a few different machine operations. The special purpose processor is not an expert at all numerical computations. It has only a few operations implemented efficiently.

2.3 Requirements Arising from Software Issues

Money and effort spent on a new power system application are usually spent in the software domain [5]. Certainly we must keep the software costs low when dealing with special purpose processors, or the inital appeal of cheaper hardware is lost. There are currently no commercially available compilers capable of taking an algorithm written in a sequential language such as Fortran and producing a language that has parallelism and pipelining, and is suitable for efficient execution on an array processor.

- A very detailed understanding of an application is necessary for translation from one computer system to another. If the new system is radically different from the old system, as it is when converting to a special purpose processor system, this requirement is very important.

- The computation placed on the special purpose processors should be in the form of small compute-intensive loops. The larger the routines that reside on the special purpose processors, the more code, time, and memory will be required.

- The algorithms should consist of fixed routines. Special purpose processors create an environment not easily changed, and very few development tools are available.

- Consideration must be given to the architecture. Otherwise all potential speed-ups may be lost. The programmer must be familiar with details of the machine such as pipelining, memory size, and memory performance.

3. Requirements That Conflict With Power System Applications

3.1 High Level Algorithm Conflicts

The main reason for not applying special purpose processors to power system computations is the impossibility of performing most of the computations on the special purpose processor. This is due to the fact that many of the computations (such as network modelling, interactive communication with the user, real-time data collection, and CRT driving) are out of the range of special purpose numerical processors. For example, in some network simulations each time-step iteration requires substantial logic and data to model each node. This can be difficult and time-consuming on a numerical special purpose processor. The alternative is to perform these tasks on the host, which is more capable of the computations. This not only places a greater burden on the host and increases communication time, but may also cause a bottleneck in the host.

Another cause of this is the lack of compute-intensive subroutines. The most compute-intensive routine is the solution of a sparse system of equations. This routine occupies approximately 1/3 [2] to 1/2 [4] of the total compute time on present systems. Unfortunately, even if we were to build a sparse simultaneous equations solver that could operate in zero time, giving no consideration to interface time, the maximum speed-up achievable would be 2. Thus, if a power study on a mainframe takes 15 minutes, it would still take at least 7.5 minutes on a mainframe with a special purpose processor.

Additionally, the computations have all been designed in a sequential manner. There may be ways to express the solution so that more than one machine can compute a solution effectively, but no such algorithms are currently employed. The same problem occurs on a lower level when mapping computations onto the architecture of the special purpose processor.

3.2 Host and Communication Requirements

The host machine will have to perform many non-numeric tasks, such as maintaining large amounts of data. A mini-computer will not suffice for the host of a special purpose processor. Tasks not performed on the special purpose processor will execute comparatively slowly on a mini-computer host, and the special purpose processor will spend too much time waiting. On the other hand, a too-powerful host will be idle while its special purpose processor is busy. When we incur the cost of a large host machine we lose some of the appeal of the cheaper special purpose processors.

Communication of information is necessary when two processors are co-operating to solve a problem. The amount of information is dependent on the algorithm and the configuration. Most system configurations involving special purpose processors require that each processor have separate memory. Since most general purpose machine manufacturers do not support the sharing of memory with foreign devices, peripheral interface remains the only method of communicating.

Long interface time between the special purpose processor and the host is due mainly to the speed of the host bus. Large data transfer may be necessary due to formatting required by one processor or the other. There may be data whose structure cannot easily be passed from host to slave, or it may be that fixed word sizes restrict all pieces of data to require the same amount of communication.

Time for transmission of data to and from a special purpose processor, activation time required to give a command to the special purpose processor, and time for the host to process the completion signal from the special purpose processor amount to considerable overhead for the host. Scheduling of the special purpose processor will also require host time. The increase in context swapping due to the new slave device may also cause a bottleneck in the host.

3.3 Hardware Conflicts

Significant hardware problems result from the irregular problem domain found in power system applications. Signal processing code can store and operate on data very regularly and control the solution with very few logical operations. Power system networks, on the other hand, are very irregular and sparse. A major characteristic of power system algorithms is the minimization of storage requirements and arithmetic operations through the use of tables, pointers, and logical operations. When sparsity is incorporated efficiently into the solution, most computations are directed by tables and most memory references are indirect.

When memory is pipelined (or interleaved), indirect addressing can be fairly expensive, since the machine must operate in non-pipeline mode. In many cases indirect addressing can dominate the computational speed of a special purpose processor. Often the total time spent in an iteration of an inner loop is dependent not on the number of operations by the ALU, but the number of memory accesses.

Another weakness of most special purpose processors is the lack of memory space. Power system applications require substantial memory to describe the simulated system. There may be up to 50 tables accessed during a simulation, some as large as 10 kilobytes. The total amount of memory used in a power system simulation is in the multi-megabyte range [8].

Flexible control of precision is desirable. Although the amount of precision required for temporary values is very large, the user may not always want to pay the price of computing in high precision. Although input to many routines may be in 4 bit precision, temporary values may require at least 32 bit precision. Internal datapaths may sometimes provide additional precision by performing intermediate calculations in a higher precision than memory provides. In addition the user can account for the lack of precision in her code by scaling the problem, but that allows more ways to commit errors.

Low level computations play an important role in obtaining efficient solutions. Sequences of computations and logical tests, if used properly, can be the key to obtaining a high level of efficiency. The programmer must pay special attention to logical calculations because they can interfere with pipelining. Also the layout of functional units and their connection for communication must be reflected in the choice of algorithms.

3.4 Software Requirements

Once we fix the hardware, the architecture, and the algorithm, we can then turn our attention to producing good software. In converting from one system to another, the best speed-up is never realized by simple program translation. (This is especially true of conversion to systems involving special purpose processors.) Data structures, algorithms, and architectures must be considered together to produce an efficient implementation on a new configuration. In most situations both careful analysis and intuition are required to determine where a speed-up can be achieved or overhead avoided. The computer system engineer knows that even a small modification can sometimes substantially improve or degrade the performance of a system.

To keep the special purpose processor busy the programmer will have to learn a new instruction set and a new architecture. When moving a problem to a special purpose processor, she must re-solve the problem and re-organize the code. This is because present systems are too large to permit a direct translation from one machine to another and Fortran doesn't express the parallelism that the problem solver knows is there. The rewritten system will not divide the work into special purpose processor operations and host operations by assigning each operation to the machine which computes it best. The computations are too throughly intermixed and too much time would be spent communicating. However, the code can be restructured so that it will be somewhat optimized for a given configuration. This requires an intimate knowledge of both the solution and the machine. Unfortunately this knowledge is fragmented over the industry. The new system will be developed in an environment that does not have a quality high level language compiler, a diagnostic compiler, library support, a simulator, or a cross compiler.

The amount of data required to make good use of the special purpose processor is larger than can easily be placed on the special purpose processor. The programmer will have to concern herself with overlays in order to fit both the program and data into memory. This requires the handling of virtual memory at the programmer level. This has been found to generate many errors and has thus been incorporated into the operating system of most conventional machines. In this situation the user is changing roles from one who uses the computer as a computational instrument to one who uses the computer to explicitly organize and manage data as well.

The programmer wants the special purpose processor to be almost invisible. The only noticeable aspect of a special purpose processor should be the speed-up of the solution. This implies that the programmer must work at keeping the special purpose device transparent to the user, but make efficient use of it. However, these two goals may not be compatible. Treating the device as a slave which executes small subroutines will involve substantial host-slave communication. Performing large sections of computation with each subroutine invocation will decrease the amount of communication per computation, but may require the restructuring of the solution (by the user) and the moving of large pieces of code to the special purpose processors (by the system programmer).

The major software hurdle does not stem from the code directly, but rather from the lack of people skilled in system conversion. Because there is an imbalance between effort required for software development and hardware development, software people are in greater demand. Software houses are likely candidates for converting to a new system. However, since the code is special purpose and good for only a special piece of hardware, the code is of very limited utility and its cost could be prohibitive. In addition to the development cost, there will be a high maintainance cost. All changes, e.g. host operating system updates, interface protocol revision, or new algorithmic developments, require the maintainer to be intimately familiar with the system.

4. Some Future Work

As we gain experience working with a certain architecture, we evolve algorithms suitable to the architecture. For example, power system algorithms that were performed by human computers 35 years ago were nearly all numerical computation. By gaining a better understanding of the machine's capabilities, the computation has changed to be approximately half numerical computation and half logical computation by using table driven sparse addressing or forms of symbolic factorization.

As we gain experience implementing power system computations on special purpose processors, the algorithms will again take on a different appearance. Consider a situation where the host must use the data to make a decision for the special purpose processor as to whether a different strategy should be employed. This could be changed to a situtation where the host "listens" to the special purpose processor and interrupts the device only when a change of strategy is in order; otherwise the special purpose processor simply continues its computations.

Presently, interactive systems do not fit well into a configuration that involves a special purpose processor because of the required communication between user and host and between host and special purpose processor. Presumably the special purpose processors can try many options faster than the user can communicate a single request. Therefore we might consider a method of interaction where the special purpose processor searches alternatives and reports the best option to the user.

5. Desirable Special Purpose Processors

Scientific algorithms are changing, so a programmable machine is necessary. There are too many different functions in most scientific computations to consider putting all of them into hardware, so a general numerical machine is necessary. Similarly, large databases must be accessed and there is no fixed data flow, so a general memory structure is necessary. The array processor is the only currently available special purpose processor with all these attributes.

An area that is not compute bound and has many years invested in its present code has very little motivation to move the code from one machine to another. When the effort is very high to achieve an efficient solution that is not (absolutely) necessary, it is not likely to be undertaken. The least effort is required to use a special purpose processor when present software works well on a special purpose processor. The power system applications are in need of a fast Fortran-like machine with large program memory, large data memory, and virtual memory. The machine cannot be only a vector machine. It must also execute scalar operations efficiently. Boundary value problems and irregular grid problems will not execute efficiently on a strong vector machine. The machine must efficiently implement a variety of operations; otherwise the poorly implemented operations will become the bottleneck of the system. In addition, the manufacturer should promise to provide future machines which are upward compatible; that is, the language should be supported through many generations of the special purpose processors. An organization cannot afford to change its programming language with every version of a special purpose processor.

6. Conclusion

Real-time simulation and graphic systems have successfully used special purpose processors. The algorithms used (such as co-ordinate transformation) have small inner loops and are real-time compute bound. There is no other way to achieve the computational speed required by the problem. To develop the system the programmer must become very intimate with the machine because she will need all of the power available from the special purpose processors.

Other areas such as seismology, computerized x-ray technology, and modal analysis have successfully used special purpose processors. The algorithms that they use (such as spectral analysis, digital filtering, correlation, and convolution) have small inner loops. They are data compute-bound. The algorithms are fixed because the systems put in the field will remain there until they are obsolete.

Areas such as structural analysis, electric power simulations, theoretical chemistry, and weather forcasting have not used special purpose processors effectively. The algorithms in each area are constantly changing. The present solutions are full of modelling issues which are always being revised. Although each area could use more compute time, in general they are not compute-bound.

There is no single issue which prohibits the use of special purpose processors in power system applications. The problems are: that the special purpose processor does not fit easily into present solution schemes, that there is a lack of small compute-intensive routines, that there are only sequential algorithms in use, that communication costs are high, and that memory accessing schemes are irregular. Therefore the software effort required will be very large and the anticipated improvement will be small. The applications could be redesigned to include the requirements necessary for efficient use of the special purpose processor [7]. But the problems associated with using special purpose processors are presently too great in relation to the computational enhancements achievable to consider restructuring the application. The power system industry should wait for technology to supply better, faster general purpose processors and for the computer industry to create an easier-to-use special purpose processor before redesigning its applications.

7. Acknowledgements

In attempting to understand and appreciate the computational problems faced by power system practitioners I had enlightening and helpful conversations with the following people: Ben Denbart (Boeing Computer Services), Ted Kostyniak and John Undril (Power Technologies Inc.), John Lamont (EPRI), Frank Orem (BPA), Robin Podmore (ESCA), Bill Tinney, and Sudhir Virmonte (Systems Control Institute).

In addition, H. T. Kung, Philip Lehman, Mike Foster, and Dave Carpenter provided helpful comments on early versions of this paper.

References

1. H. H. Happ, C. Pottle, and K.A. Wirgau. An Assessment of Computer Technology for Large Scale Power System Simulation. 1979 Power Industry Computer Applications Conference Proceedings, IEEE, 1979, pp. 316-324.

2. F.M. Orem and W.F. Tinney. Evaluation of an Array Processor for Power System Applications. 1979 Power Industry Comuter Applications Conference Proceedings, IEEE, 1979, pp. 345-350.

3. N. S. Ostlund. Attached Scientific Processors for Chemical Computations: A Report to the Chemistry Community. National Resource for Computation in Chemistry, to appear (1979).

4. R. Podmore, N. M. Peterson, M. Liveright, J. Britton, and S. Virmani. Application of an Array Processor for Power System Network Computations. 1979 Power Industry Computer Applications Conference Proceedings, IEEE, 1979, pp. 325-330.

5. R. Podmore. Array Processor Applications in System Operation. Presented before the Parallel Processing for Power System Planning and Operations Workshop, Dallas, Texas, October 1980.

6. C. Pottle, M. S. Pottle, R. W. Tuttle, R. J. Kinch, and H. A. Scheraga. Conformational Analysis of Proteins: Algorithms and Data Structures for Array Processing. Presented before the Division of Computers in Chemistry at the 179th National Meeting of the American Chemical Society, Houston, Texas, March 1980.

7. H. S. Stone. Computer Architecture in the 1980's. Computer Science and Scientific Computing, Institute for Computer Applications in Science and Engineering, 1976, pp. 255-282.

8. J. M. Undril and T. E. Kostyniak. Experience with an Array Processor. Presented before the Parallel Processing for Power System Planning and Operations Workshop, Dallas, Texas, October 1980.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-81-108	2. GOVT ACCESSION NO. AD-A106552	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EFFECTS OF SPECIAL PURPOSE HARDWARE IN SCIENTIFIC COMPUTATION WITH EMPHASIS ON POWER SYSTEM APPLICATIONS		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) DIANE M. DETIG		8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0236
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Department Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE November 1980
		13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LME